IMPLEMENTASI FRAMEWORK SPRING MVC UNTUK PEMBUATAN SISTEM INFORMASI MANAJEMEN E COMMERCE

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat Mencapai Gelar Ahli Madya
Program Diploma III Ilmu Komputer

Diajukan Oleh:
MUDZAKKIR TOHA
NIM. M3107036

PROGRAM DIPLOMA III ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS SEBELAS MARET
2010

HALAMAN PERSETUJUAN

IMPLEMENTASI FRAMEWORK SPRING MVC UNTUK PEMBUATAN SISTEM INFORMASI MANAJEMEN E COMMERCE

Disusun Oleh:

MUDZAKKIR TOHA
NIM. M3107036

Tugas akhir ini telah disetujui untuk dipertahankan
di hadapan dewan penguji pada tanggal 12 Juli 2010

Pembimbing Utama

Wiharto,S.T. M. Kom
NIP. 197502102008011005

HALAMAN PENGESAHAN

IMPLEMENTASI FRAMEWORK SPRING MVC UNTUK PEMBUATAN SISTEM INFORMASI MANAJEMEN E COMMERCE

Disusun Oleh:

MUDZAKKIR TOHA
NIM. M3107036

Dibimbing oleh:
Pembimbing Utama

Wiharto,S.T. M. Kom
NIP. 197502102008011005

Tugas akhir ini telah diterima dan disahkan oleh dewan penguji tugas akhir Program Diploma III Ilmu Komputer

pada hari Senin tanggal 12 Juli 2010

Dewan Penguji:

1. Penguji 1	Wiharto,S.T. M. Kom NIP. 197502102008011005	(_____)
2. Penguji 2	Sri Arum, S.Z, S.Kom NIP.	(_____)
3. Penguji 3	Agus Purnomo, S.Si NIP.	(_____)

Disahkan Oleh:

a.n.Dekan Fakultas MIPA Pembantu Dekan I	Ketua Program Diploma III Ilmu Komputer
Ir. Ari Handono Ramlan, M.Sc, Ph.D NIP. 19610223 198601 1 001	**Drs. Y.S Palgunadi, M.Sc.** NIP. 19560407198303 1 004

ABSTRACT

Mudzakkir Toha. 2010. **IMPLEMENTASI FRAMEWORK SPRING MVC UNTUK PEMBUATAN SISTEM INFORMASI MANAJEMEN E COMMERCE. The Implementation of Spring MVC Framework to Create an E Commerce Information Management System**. Computer Science Pregraduate Program. Information Engineering. Mathematics and Natural Sciences Faculty. Sebelas Maret University.

Object oriented technology is recent of analysis in application software development computer based. The old way of software development is structured programming, that is not reusable.

One way of modeling technology object oriented programming is using UML (Unified Modeling Language). UML have become standard language for modelling of object oriented system in world. MVC architecture will assist of application maintaining. MVC architecture caused application more easy and structured.

Has created an application that implementing the MVC concept.

Keyword : JEE 5, Spring MVC, Hibernate JPA.

ABSTRAK

Mudzakkir Toha. 2010. **IMPLEMENTASI FRAMEWORK SPRING MVC UNTUK PEMBUATAN SISTEM INFORMASI MANAJEMEN E COMMERCE.** Program DIII Ilmu Komputer. Teknik Informatika. Fakultas Matematika dan Ilmu Pengetahuan Alam. Universitas Sebelas Maret Surakarta.

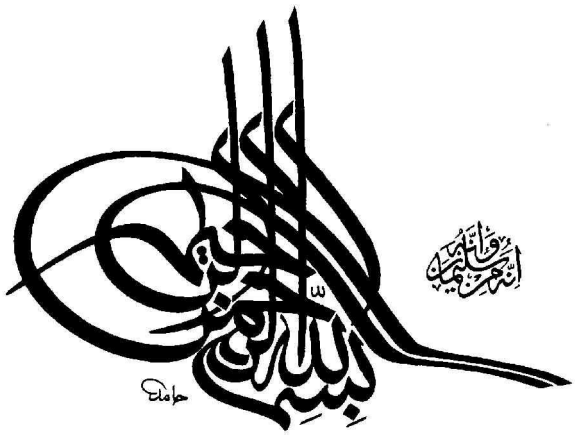
Teknologi objek merupakan pendekatan analisis termutakhir dalam pengembangan perangkat lunak aplikasi berbasis komputer. Cara pengembangan perangkat lunak yang lama adalah pemgroraman terstruktur, yang ternyata dalam implementasinya software yang dibuat dengan pemrograman terstruktur tidak *reusable*.

Salah satu cara untuk memodelkan teknologi pemrograman berbasis objek adalah dengan menggunakan UML (*Unified Modeling Language*). UML sudah menjadi bahasa standar untuk pemodelan sistem berorientasi objek di dunia. Arsitektur MVC akan membantu memudahkan perbaikan aplikasi. Arsitektur MVC menyebabkan aplikasi lebih terstruktur dan mudah.

Terciptalah aplikasi yang mengimplementasikan konsep MVC.

Kata kunci : JEE 5, Spring MVC, Hibernate JPA

HALAMAN MOTTO



Rabb, hidupkanlah aku jika hidup itu baik untukku..

Matikanlah aku, jika mati itu baik untukku.

HALAMAN PERSEMBAHAN

Dedicated to :

krissadewo.wordpress.com, terima kasih bimbingannya

jasoet.wordpress.com, terima kasih nasehatnya

Developer java yang telah berbagi ilmunya di blog masing-masing, yaitu : ifnu.artivisi.com, endy.artivisi.com, martinusadyh.web.id, eecchhoo.wordpress.com, loianegroner.com

Teman-teman juggers Indonesia, dan juggers jug-joglosemar, terima kasih telah berbagi

KATA PENGANTAR

Assalamu'alaikum Warohmatullohi Wabarokatuh.

Bismillahirrohmanirrohim, segala puji dan rasa syukur hanya penulis panjatkan ke haribaan Alloh *subhanahu wa ta'ala,* yang telah melimpahkan segala kemudahannya hingga akhirnya penulis mampu menyelesaikan Tugas Akhir dan menuliskan laporannya tepat waktu.

Laporan Tugas Akhir ini disusun untuk memenuhi sebagian persyaratan memperoleh kelulusan Diploma III Teknik Informatika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Sebelas Maret Surakarta. Dalam pelaksanaan Tugas Akhir, yang didalamnya termasuk kegiatan pembuatan laporan ini, penulis mendapat banyak bantuan dari berbagai pihak. Tanpa bantuan Alloh *subhanahu wa ta'ala* melalui tangan mereka niscaya Tugas Akhir penulis tidak akan berjalan dengan lancar. Untuk itu dalam secuil kertas yang mungkin tiada berarti ini penulis sampaikan rasa hormat dan menghaturkan rasa terima kasih kepada:

1. YS. Palgunadi, M.Sc., selaku Ketua Program Diploma III Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Sebelas Maret Surakarta.
2. Wiharto,S.T. M. Kom, selaku pembimbing, yang telah benar-benar membuka mata penulis akan ilmu-ilmu yang sempat tidak terpandang oleh penulis.
3. Muhammad Syafi'i, S.Si., selaku dosen yang menangani Tugas Akhir, terima kasih.
4. Teman-teman Java User Group Joglosemar yang senantiasa mensupport developer Java wilayah Jogja, Solo dan Semarang.

Bukan lagi rahasia, sebuah karya selalu disertai kekurangannya, oleh sebab itu penulis memohon kelapangan hati pembaca sekalian untuk menerima kekurangan yang ada dalam laporan Tugas Akhir ini.

Wassalamu'alaykum Warohmatullohi Wabarokatuh.

Surakarta, 10 Juni 2010

Penulis

DAFTAR ISI

Halaman

HALAMAN JUDUL i
HALAMAN PERSETUJUAN ii
HALAMAN PENGESAHAN iii
ABSTRACT iv
ABSTRAK v
HALAMAN MOTTO vi
HALAMAN PERSEMBAHAN vii
KATA PENGANTAR viii
DAFTAR ISI x
DAFTAR GAMBAR xii
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Perumusan Masalah 1
1.3 Batasan Masalah 1
1.4 Tujuan dan Manfaat 2
1.5 Metodologi Penelitian 2
1.6 Sistematika Penulisan 2
BAB II LANDASAN TEORI 4
2.1 Bahasa Pemrograman JAVA 4
2.2 Java Enterprise Edition (Java EE) 5
2.3 Spring Framework 5
2.4 Pengenalan Arsitektur MVC 8
2.5 ORM (*Object Relational Mapping*) 9
2.6 DAO 9
2.7 *Three Tier* 9
2.8 *E Commerce* 10
BAB III DESAIN DAN PERANCANGAN 11
3.1 *Use Case Diagram* 11

3.2 *Activity Diagram* 12
3.3 *Sequence Diagram* 12
3.4 *Class Diagram* 13
3.5 Arsitektur *Three Tier* Spring, Hibernate, Tomcat dan PostgreSQL.. 13
3.6 Arsitektur MVC JSP, Spring MVC dan Hibernate JPA 14
BAB IV IMPLEMENTASI DAN ANALISA 15
4.1 Implementasi 15
4.2 Analisa 32
BAB V PENUTUP 33
5.1 Kesimpulan 33
5.2 Saran 33
DAFTAR PUSTAKA 34

DAFTAR GAMBAR

No

1. Arsitektur Spring *Framework* 7
2. *Use Case Diagram* 11
3. *Activity Diagram* 12
4. *Sequence Diagram* 12
5. *Class Diagram* 13
6. Arsitektur *Three Tier* 13
7. Arsitektur MVC 14
8. Halaman Home 15
9. Halaman Tambah Barang 16
10. Halaman Daftar Barang 17
11. *Form* Tambah Costumer 17
12. *Form* Tambah Faktur 18
13. *Form* Tambah Item Barang 18
14. *List* Faktur 19
15. *Report* Faktur 19
16. Daftar Admin 20
17. Daftar Kasir 21
18. Daftar Petugas 22
19. Konfigurasi *Database* 22
20. Implementasi Penggunaan PostgreSQL 23
21. *Server* Aplikasi Tomcat 24
22. Persistence Unit 24
23. Annotasi *Entity* Menandakan *Entity* Model 25
24. BarangDAO 26
25. Implementasi BarangDAO 26
26. Konfigurasi Dispatcher Servlet web.xml 27
27. Konfigurasi Application Context web.xml 27
28. Konfigurasi *Entity* Manager Factory 28

29. *Controller* Halaman Manipulasi Model Barang 29
30. Halaman Daftar Barang Contoh JSP 30
31. Konfigurasi Sitemesh..... 31
32. *Browser* sebagai *Client Interpreter* 31

BAB I
PENDAHULUAN

1.1 Latar Belakang

Suatu program akan sulit untuk diperbaiki tanpa menggunakan konsep MVC. Hal itu dapat terjadi dikarenakan *client* dalam aplikasi yang dibuat adalah *thick client*, yaitu *client* yang juga akan menangani *business logic* sehingga proses untuk *maintenance* aplikasi tersebut membutuhkan waktu yang lama. Untuk mereduksi kelemahan tersebut, diciptakan konsep MVC (*Model View Controller*), sehingga terpisahlah 3 *layer* yang berbeda, sehingga proses *maintenance* suatu aplikasi akan semakin cepat. Dengan memisahkan antara model, *business logic*, dan *view*, aplikasi *e-commerce* akan lebih mudah untuk di *maintain*.

1.2 Perumusan Masalah

Perumusan masalah adalah: "Bagaimana mengimplementasikan model MVC ke dalam sebuah sistem informasi manajemen penjualan *online* yang berbasis Java EE versi 5, dengan menggunakan framework Spring MVC".

1.3 Batasan Masalah

Batasan masalah pada penulisan tugas akhir ini adalah implementasi model MVC pada aplikasi sistem informasi manajemen penjualan *online* yang berbasis Java EE versi 5, dengan menggunakan framework Spring MVC pada proses penerimaan transaksi, input barang, dan input petugas.

1.4 Tujuan dan Manfaat

Tujuan dari tugas akhir penulis adalah mengimplementasikan model MVC ke dalam sebuah sistem informasi manajemen penjualan *online* yang berbasis Java EE versi 5, dengan menggunakan framework Spring MVC.

Sedangkan manfaat yang diharapkan bisa diperoleh dengan adanya tugas akhir penulis ini adalah mengasah kemampuan penulis sendiri dalam

mengimplementasikan apa yang selama ini telah dipelajari, dan semoga bisa menjadi acuan bagi akademisi untuk mempelajari tentang apa yang penulis buat.

1.5 Metodologi Penelitian

Metode pengumpulan data yang akan digunakan dalam pembuatan Tugas Akhir penulis adalah studi pustaka, yaitu : Mengumpulkan data dari membaca buku dan literatur yang berhubungan dengan permasalahan yang dijadikan objek penelitian.

1.6 Sistematika Penulisan

Dalam penulisan Tugas Akhir ini penulis menggunakan sistematika penulisan sebagai berikut:

BAB I PENDAHULUAN

Bab ini menjelaskan secara umum latar belakang masalah, perumusan masalah, batasan masalah, tujuan dan manfaat, metode penelitian, dan sistematika penulisan.

BAB II LANDASAN TEORI

Bab ini berisikan landasan teoritis yang digunakan dalam pembuatan tugas akhir. Landasan teori diperoleh selama penelitian studi pustaka. Dasar-dasar teori tersebut berhubungan dengan metode pemecahan masalah yang diterapkan pada tugas akhir.

BAB III DESAIN DAN PERANCANGAN

Bab ini menerangkan perancangan dan desain sistem atau Tugas Akhir yang dibangun oleh penulis. Perancangan dan desain tersebut meliputi pemodelan proses, pemodelan basis data dan perancangan struktur navigasi

BAB IV ANALISIS DAN IMPLEMENTASI

Bab ini menjelaskan implementasi perancangan dalam hal kerja sistem berikut analisis terhadap sistem. Bab ini berisi dua subbab yaitu analisis dan implementasi. Subbab implementasi berisi penjelasan dari desain antarmuka sistem, sedangkan subbab analisis menjelaskan kinerja sistem ketika sistem telah diimplementasikan.

BAB V PENUTUP

Bab ini berisi kesimpulan yang didapatkan dari analisis mengenai keterkaitan dengan tujuan pembuatan sistem, berikut saran-saran berkaitan dengan penggunaan sistem dan atau pengembangan sistem di masa yang akan datang.

BAB II
LANDASAN TEORI

2.1 Bahasa Pemrograman JAVA

JAVA adalah sebuah bahasa pemrograman komputer berbasis pada *Object Oriented Programming*. Java diciptakan setelah C++ dan didesain sedemikian sehingga ukurannya kecil, sederhana, dan *portable* (dapat dipindah – pindahkan di antara bermacam platform dan sistem operasi). Program yang dihasilkan dengan bahasa Java dapat berupa APPLET (aplikasi kecil yang jalan diatas *web browser*), MIDlet (aplikasi yang berjalan di ponsel) maupun berupa aplikasi mandiri yang dijalankan dengan program java *Interpreter*.

Salah satu keunggulan Java adalah sifatnya yang *platform independence*, artinya baik *source* program maupun hasil kompilasinya sama sekali tidak bergantung kepada sistem operasi dan platform yang digunakan. *Source code* sebuah aplikasi dengan bahasa Java yang ditulis diatas sistem Windows NT misalnya, dapat dipindahkan ke sistem operasi UNIX tanpa harus mengubah satu baris kode pun. Ini tentunya merupakan satu nilai tambah tersendiri.

Generasi yang saat ini sedang berkembang dari *platform* Java ialah Java 2. Agar sebuah program Java dapat dijalankan, maka file dengan ekstensi Java harus dikompilasi menjadi *file bytecode*. Untuk menjalankan *bytecode* ini, dibutuhkan JRE (*Java Runtime Environment*) yang memungkinkan pemakai untuk menjalankan program Java, hanya menjalankan, dan tidak untuk membuat kode baru lagi. JRE berisi JVM dan library Java yang digunakan.

Java *Platform* dibagi menjadi 3 kategori, yatu :

a. *Java Standard Edition* (JSE)

   Kategori ini digunakan untuk mengembangkan dan menjalankan aplikasi Java berbasis PC.

b. *Java Enterprise Edition* (JEE)

   Kategori ini digunakan untuk mengembangkan dan menjalankan aplikasi Java pada lingkungan *enterprise*, dengan fungsi-fungsi seperti *Enterprise Java Bean* (EJB), *Servlet*, dan *Java Server Page* (JSP).

c. *Java Micro Edition* (JME)

Kategori ini digunakan untuk mengembangkan dan menjalankan aplikasi java berbasis *handled device*, seperti *Personal Digital Assistant* (PDA), *handpone*, dan *pocketPC*.

2.2 *Java Enterprise Edition* (Java EE)

Java EE adalah *platform* terdepan untuk pengembangan dan penggunaan aplikasi web dan *enterprise*. Java EE didefinisikan dengan spesifikasi. Seperti dengan spesifikasi *Java Community Process*, Java EE informal juga dianggap menjadi standar sejak selular harus sepakat untuk *conformance* persyaratan tertentu untuk menyatakan mereka sebagai produk Java EE *compliant*, walau tanpa ISO atau standar ECMA. *Java Enterprise Edition* (Java EE) Class juga merupakan salah satu *Java Family Suite*, yang menjadi standard penting untuk mengembangkan aplikasi *enterprise multitier* berbasis komponen. Diantaranya adalah untuk aplikasi *e-bussiness*, *e-commerce* dan *web based application*.

2.3 Spring Framework

Spring adalah salah satu *application framework* untuk aplikasi berbasis Java, tepatnya JEE. Spring merupakan sebuah *framework* (kerangka kerja) yang digunakan untuk membangun sebuah aplikasi *Enterprise*. Spring termasuk *framework* yang *lightweight* (ringan) untuk mendukung secara penuh dalam pengembangan aplikasi *Enterprise* siap pakai.

Spring dapat digunakan untuk melakukan pengaturan deklarasi manajemen transaksi, remote access dengan menggunakan RMI atau layanan web lainnya, fasilitas *mailing*, dan beragam opsi untuk pengaturan data ke database. Spring juga memungkinkan kita menggunakan hanya modul-modul tertentu sehingga kita tidak usah menggunakan semua modul spring dalam aplikasi apabila tidak diperlukan.

2.4 Pengenalan Arsitektur MVC

Fitur-fitur dari Spring Framework :

1. *Transaction Management* : Spring framework menyediakan sebuah layer abstrak yang generik untuk manajemen transaksi, sehingga memudahkan para developer dalam melakukan manajemen transaksi.
2. *JDBC Exception Handling* : layer abstrak JDBC menawarkan *exception* yang bersifat hierarki sehingga memudahkan penanganan error.
3. *Integration with Hibernate, JDO, and iBatis* : Spring menawarkan layanan integrasi terbaik dengan Hibernate, JDO dan iBatas
4. *AOP Framework* : Spring merupakan *framework* AOP Terbaik yang pernah ada.
5. *MVC Framework* : Spring hadir dengan *framework* aplikasi web MVC, yang dibangun di atas inti Spring. Spring merupakan *framework* yang sangat fleksibel dalam pengaturan strategi interface, dan mengakomodasi beberapa teknologi *view* seperti JSP, Velocity, Tiles, iText, dan POI.

Arsitektur Spring :

1. Spring AOP

   Salah satu komponen utama Spring adalah AOP *framework*, AOP *framework* digunakan untuk :

   a. Untuk menyediakan layanan *Enterprise*, terutama sebagai pengganti EJB. Layanan terpenting dalam layanan ini adalah untuk mendekralif manajemen transaksi, yang telah disediakan dalam abstraksi *spring transaction*.

   b. Untuk memungkinkan pengguna dalam menerapkan AOP dalam penggunaan OOP.

2. Spring ORM

   Spring ORM berhubungan dengan akses database dan menyediakan lapisan layer terintegrasi dengan ORM yang populer termasuk JDO, Hibernate dan iBatis.

3. Spring Web

Merupakan bagian dari modul pengembangan Web Spring termasuk Spring Web MVC.

4. Spring DAO

   DAO (*Data Access Object*) mendukung standarisasi akses data yang menggunakan teknologi seperti JDBC, Hibernate dan JDO.

5. Spring Context

   Paket ini didasari pada paket beans untuk menambah dukungan sumber pesan dan untuk pola desain Observer, dan kemampuan untuk mendapatkan sumber daya yang konsisten dengan menggunakan API.

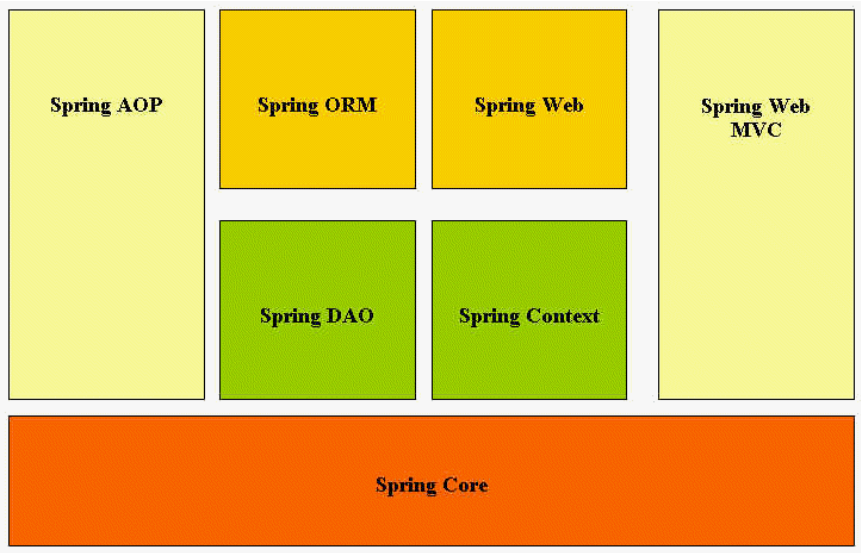
6. Spring Web MVC

   Menyediakan implementasi MVC untuk aplikasi web.

7. Spring Core

   Paket Spring Core ini merupakan komponen paling penting dari Spring Framework. Komponen ini menyediakan fitur *Dependency Injection*. *BeanFactory* memisahkan dependensi seperti inisialisasi, pembentukan dan akses object dari logika program anda.

   Diagram Berikut menggambarkan arsitektur dari Spring :



Gambar 2.1. Arsitektur Spring *Framework*

Pola desain MVC (*Model-View-Controller*) memberikan pemecahan permasalahan *coupling* yang tinggi tersebut dengan men-*decoupling* lapisan *data access*, *business logic*, dan *data presentation* atau *user interaction*.

a. Model

Model merepresentasikan lapisan data *enterprise* dan logika atau rule bisnis yang akan mengakses dan mengupdate data tersebut. Pada bagian model juga merepresentasikan proses riil yang terjadi pada suatu objek. Bisa termasuk dalam bagian ini adalah *java beans* dengan properti dan method yang dimiliki (*getters, setters dan constructors*). Javabeans berhubungan dengan konsep kegunaan dari *website* yang kita buat yang akan digunakan oleh user. Sebagai contoh, jika web yang dikembangkan adalah tentang penjualan produk, maka tentu akan terdapat beberapa *java beans* inti seperti : produk, customer, order, *invoice*. Secara singkat dapat kita sebut bahwa bagian model merupakan inti dari *layer* bisnis.

b. View

Bagian ini bisa dibangun dengan teknologi JSP. Bagian ini yang akan memberikan tampilan kepada user, dari data yang didapatkan dari lapisan model. Jadi JSP digunakan disini hanya untuk menampilkan data saja. Bagian ini bisa juga disebut *presentation layer*. Selain JSP bisa juga dengan menggunakan JSF

c. Controller

Bagian ini digunakan untuk menerima setiap request dan memformulasikan sutau response untuk request tersebut. Ini bisa dilakukan dengan teknologi *servlet*, yang bisa juga diimplementasikan dengan JSP atau JSF.

Arsitektur *Model-View-Controller* adalah sebuah pola yang terbukti membangun proyek secara lebih efektif. Hal itu dilakukan dengan memilah komponen antara *Model, View* dan *Controller* pada bagian-bagian dalam proyek.

2.5 ORM (Object Relational Mapping)

ORM adalah sebuah teknologi yang menjembatani antara paradigma pemrograman berorientasi objek dengan database relational.

ORM memandang suatu aplikasi sebagai suatu himpunan object (*entity / value*) yang memainkan peran-peran (bagian dalam hubungan-hubungan). Kadang disebut sebagai modeling berbasis fakta karena dalam ORM menyatakan secara lisan relevan data sebagai fakta-fakta dasar (*elementary fact*) yang dapat dipecah menjadi fakta-fakta kecil tanpa adanya informasi yang hilang. ORM sudah digunakan dalam 3 dekade dan sekarang mempunyai dukungan tools industrial modelling, namun masih belum memiliki *metamodel* resmi yang standar.

2.6 DAO

Data access object atau lebih terkenal singkatannya yaitu DAO merupakan *design pattern* yang biasa digunakan oleh seorang *java developer* dalam membangun sebuah sistem berbasis database. DAO merupakan sebuah konsep dimana digunakan untuk menangani kasus yang terjadi dalam bisnis *logic*, atau lebih gampangnya proses yang berhubungan dengan manipulasi data dalam *database*. DAO merupakan pola membangun sebuah bisnis *logic* secara terstruktur sesuai dengan entitas yang terdapat pada *database*.

2.7 Three Tier

Pada awalnya *client*/*server* merupakan sebuah sistem yang berhubungan dalam sebuah jaringan yang memiliki dua komponen utama yaitu *client* dan *server*. Istilah ini kemudian dikenal juga dengan “2-tier”. Kemudian istilah tier itu sendiri digunakan untuk menjelaskan pembagian sebuah aplikasi yang melalui *client* dan *server*. Saat ini, pembagian proses kerja telah menjadi bagian utama dari konsep *client*/*server* sehingga proses pembagian kerja telah diatur lebih spesifik.

Two-tier membagi proses load ke dalam 2 bagian. Aplikasi utama berjalan pada sisi client yang biasanya mengirimkan request dalam bentuk *syntax* SQL ke sebuah *database server* yang fungsinya menyimpan data.

Perkembangan internet dan jaringan yang pesat saat ini tidak memungkinkan lagi diselesaikan dengan metode 2-tier *client*/*server*. Aplikasi *client*/*server* berskala luas telah dikembangkan dan kini muncul *E-Commerce* yang berbasis internet. Hal tersebut tentunya membuat aplikasi-aplikasi semakin kompleks. Aplikasi-aplikasi tersebut dibagi menjadi beberapa komponen dan didistribusikan melalui *multiprocesor*. Banyak perusahaan besar yang sudah menggunakan *client*/*server* mulai merasakan 2-tier tidak relevan lagi untuk diimplementasikan kembali. Hal ini disebabkan karena perusahaan dituntut harus dapat mendukung internet dan semua komponennya. Aplikasi tersebut harus dapat melayani ribuan komputer *client* dimana aplikasi ini sering berjalan pada banyak *server* dan terdiri dari ratusan komponen-komponen *software* di dalamnya. 3-tier membagi proses loading antara :

i. Komputer client menjalankan GUI *logic*,
ii. Aplikasi server menjalankan *business logic*, dan
iii. Database atau *legacy application*.

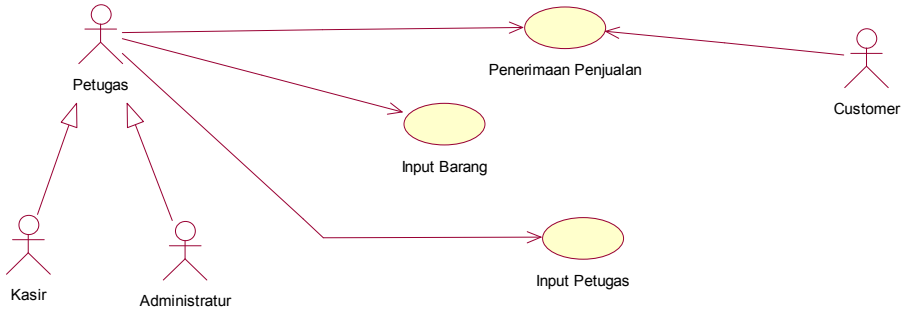
2.8 *E Commerce*

E-commerce adalah membeli atau menjual produk atau jasa melalui media elektronik, salah satunya adalah melalui media internet. Melalui *e-commerce* ini pelanggan tidak perlu lagi datang ke sebuah toko untuk membeli barang yang diinginkan tetapi pelanggan dapat secara langsung memesan barang mereka melalui internet. Di Indonesia sendiri pembelian atau penjualan melalui *e-commer*ce sudah sering digunakan. Selain lebih mudah penerapannya, dalam segi biaya juga bisa di katakan murah, berbisnis di internet juga tidak begitu menyita waktu.

BAB III
DESAIN DAN PERANCANGAN

3.1 *Use Case Diagram*

Untuk memperoleh gambaran awal sistem perlu dibuat use case diagram. Use case diagram menggambarkan ruang lingkup aplikasi akan merekam kejadian apa saja dalam dunia nyata. Use case diagram sebagaimana tergambar pada gambar *use case diagram*.

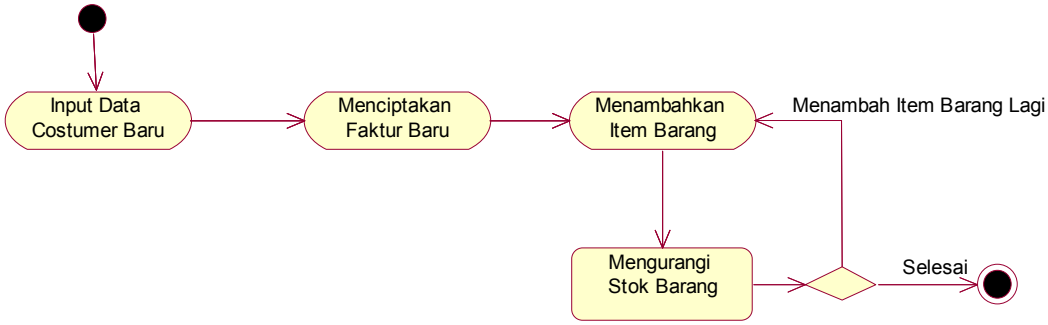


Gambar 3.1 *Use Case Diagram*

Deskripsi:

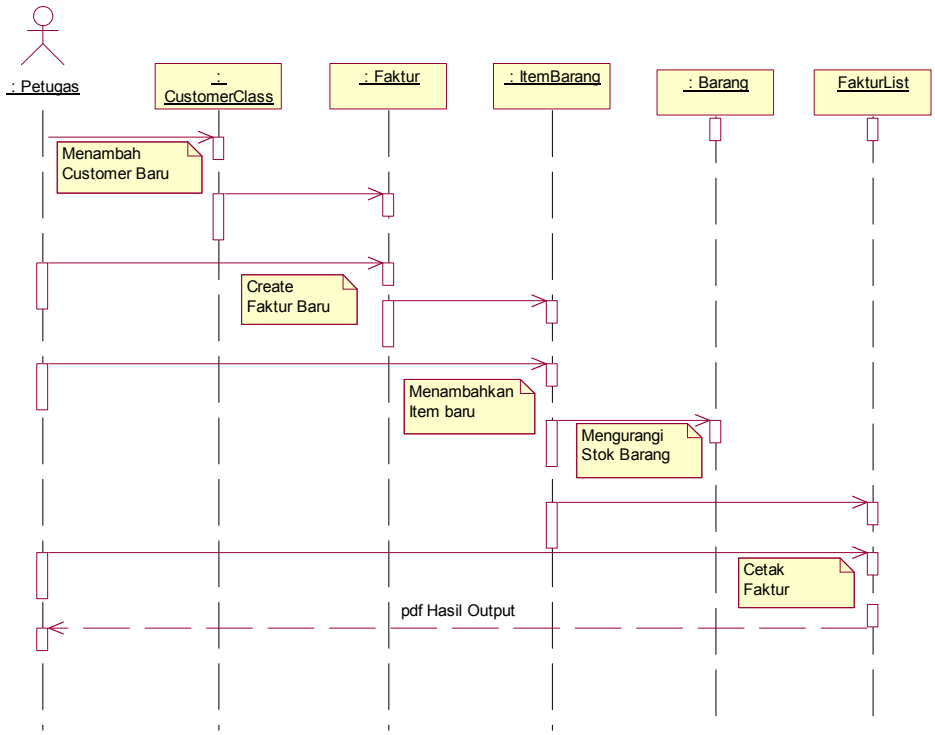
1. Penerimaan Penjualan.
   Actor : Petugas, Customer
   Deskripsi : Petugas memasukkan data transaksi. Customer menerima bukti faktur.
2. Input Petugas
   Actor: Petugas
   Deskripsi : Petugas memasukkan data petugas baru.
3. Input Barang
   Actor : Petugas
   Deskripsi : Petugas memasukkan data barang baru.

3.2 *Activity Diagram*



Gambar 3.2 *Activity Diagram*

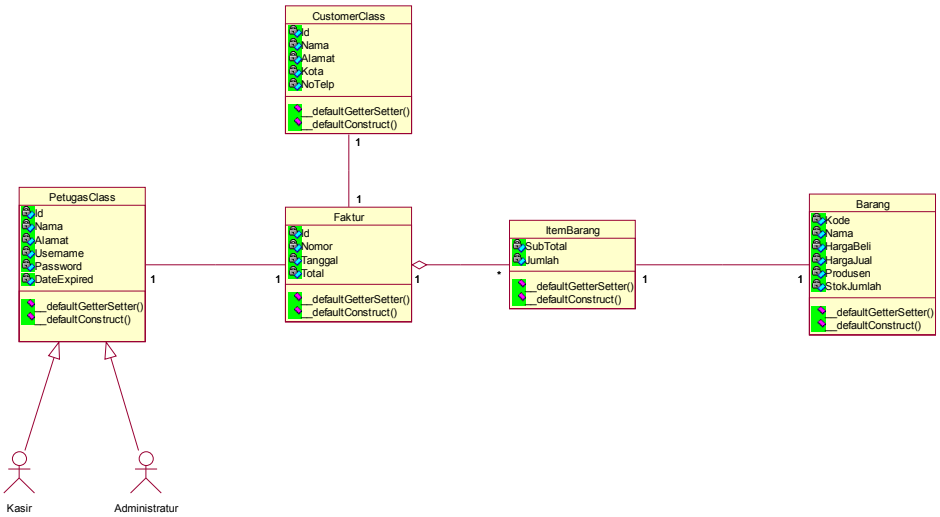
3.3 *Sequence Diagram*



Gambar 3.3. *Sequence Diagram*

3.4 *Class Diagram*

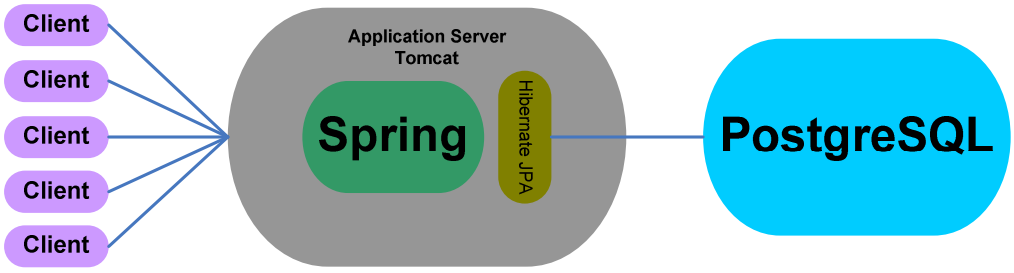
Class diagram sebagaimana tergambar pada gambar c*lass diagram*.



Gambar 3.4 *Class Diagram*

3.5 Arsitektur *Three Tier* Spring, Hibernate, Tomcat dan PostgreSQL

Arsitektur three tier sebagaimana tergambar pada gambar arsitektur *three tier*.



Gambar 3.5 Arsitektur *Three Tier*

3.6 Arsitektur MVC JSP, Spring MVC dan Hibernate JPA

Arsitektur MVC antara JSP, Spring MVC dan Hibernate JPA sebagaimana tergambar pada gambar arsitektur MVC.

Gambar 3.6 Arsitektur MVC

BAB IV
IMPLEMENTASI DAN ANALISIS

4.1 Implementasi

Implementasi perancangan terhadap sistem yang dibangun bisa dilihat melalui desain menu *home*, yang secara garis besar adalah sebagai berikut:

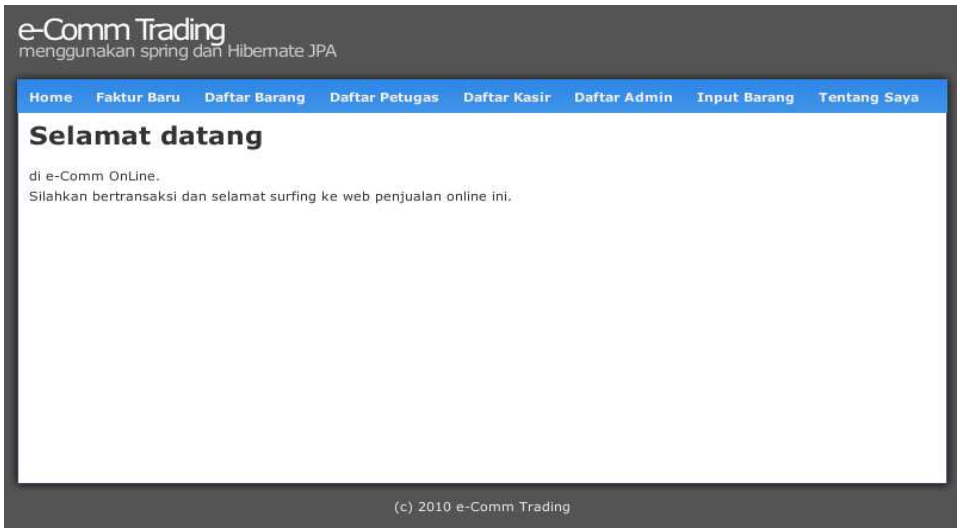
4.1.1 Implementasi Use Case Diagram

1. Halaman *Home*

Ketika pertama kali aplikasi ini di *load*, maka halaman yang pertama kali di *load* adalah halaman index. Hal ini dikarenakan pada project JEE, file yang pertama kali di *load* adalah web.xml. Dalam project kali ini file web.xml me-*redirect* ke halaman redirect.jsp, yang kemudian me-*redirect* ke halaman index.jsp.

Isi file redirect.jsp:

```
<%@taglib          uri="http://java.sun.com/jsp/jstl/core"
prefix="c"%>
<c:redirect url="index.jsp"/>
```



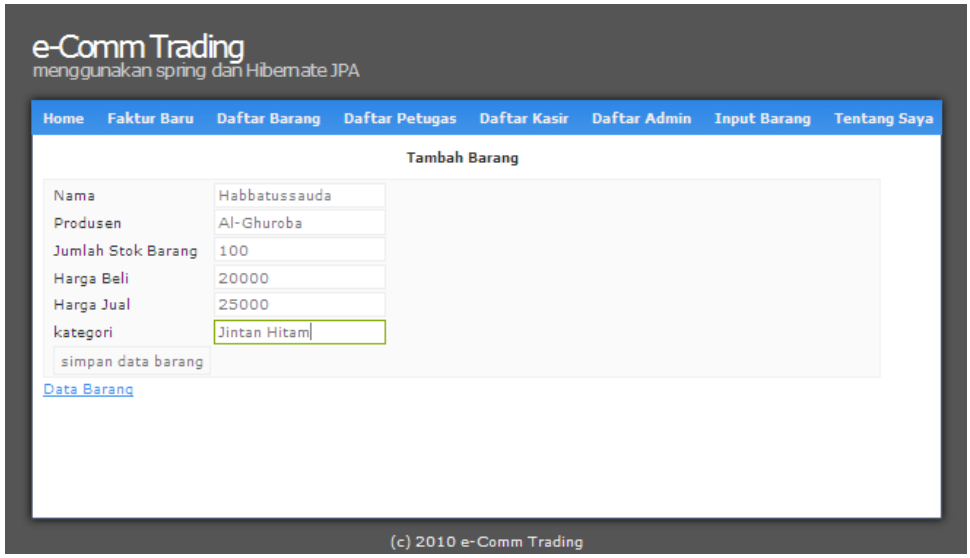
Gambar 4.1 Halaman *Home*

Dari gambar 4.1 diatas, dapat kita lihat bahwa halaman index.jsp merupakan halaman utama yang muncul pertama kali ketika program *runing*. Pada halaman ini berisikan *content* berupa tulisan

"Selamat datang", yang menjadi kata pembuka. Dari halaman *home* ini nantinya admin dapat melakukan semua proses yang diinginkan. Dibagian atas *form* halaman *home*, terdapat beberapa menu, yaitu *Home*, Faktur Baru, Daftar Barang, Daftar Petugas, Daftar Kasir, Daftar Admin, input barang, dan tentang saya.

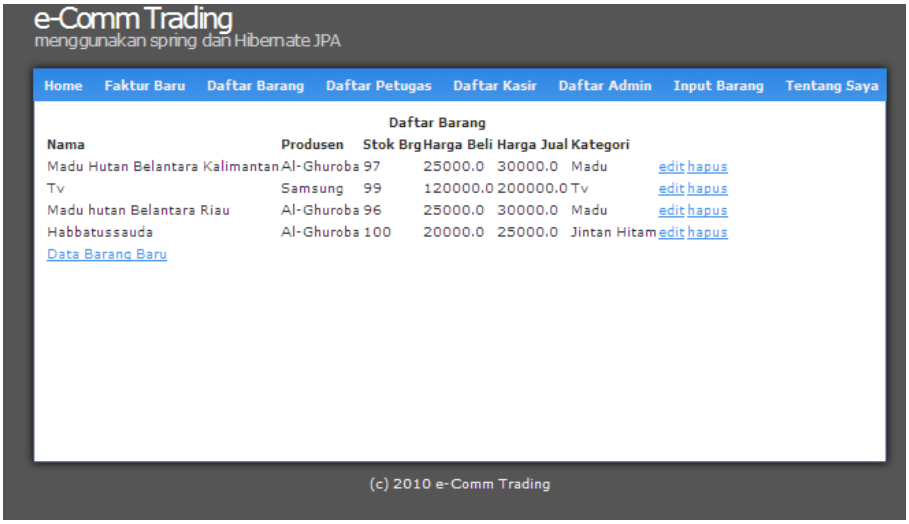
2. Menambahkan Barang Baru

Navigasi input barang digunakan untuk menambahkan barang baru. navigasi tersebut menuju halaman BarangForm.htm.



Gambar 4.2 Halaman Tambah Barang

Setelah barang di tambahkan, dan di simpan di dalam *database*, maka akan menuju halaman barangList.htm, yaitu daftar keseluruhan barang.



Gambar 4.3 Halaman Daftar Barang

3. Menambahkan Faktur Baru

Gambar 4.4 *Form* Tambah Costumer

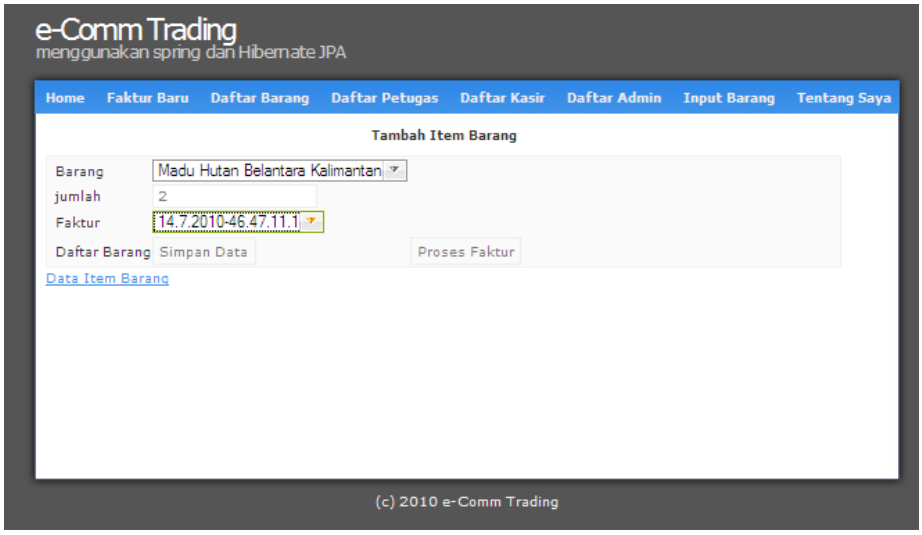
Pada halaman ini, terdapat *form* data *costumer*. Disini petugas menginputkan data dari *costumer* ke dalam *database*. Data costumer ini nantinya akan digunakan dalam proses pembuatan faktur penjualan barang. Halaman *costumer* adalah halaman yang akan di *load* pertama kali saat petugas akan menambah faktur baru.

Pada halaman tambah faktur, petugas dapat membuat data faktur. Nomor faktur berasal dari tanggal pemesanan barang, kemudian

petugas mengisikan nama *costumer*, lalu mengisi *field* petugas. Kemudian input data faktur tersebut di simpan ke dalam *database*.

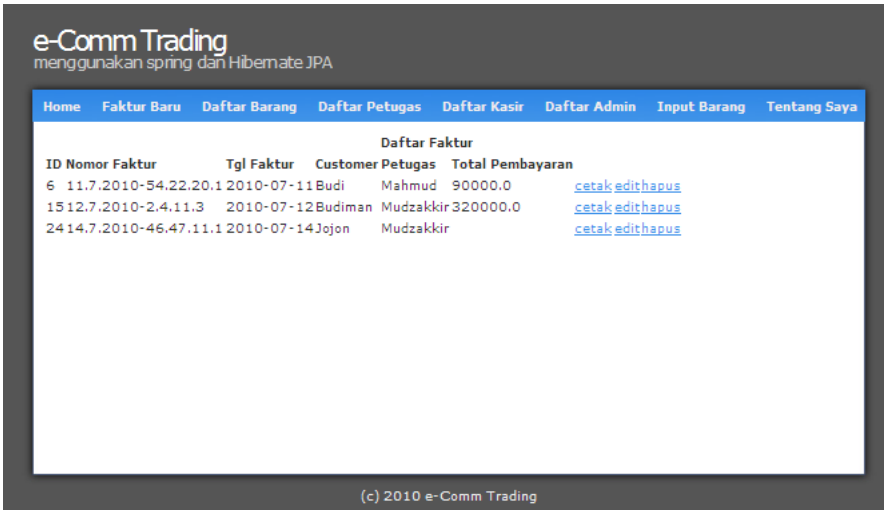
Gambar 4.5 *Form* Tambah Faktur

Pada halaman item barang, terdapat *form* untuk menambahkan item barang ke dalam faktur, yang nantinya dijadikan data faktur. Di dalam *form* ini terdiri dari nama barang yang inputannya berupa *combo box*. Selain itu juga terdapat *fileld* jumlah yang merupakan banyaknya stok barang yang tersedia. Di dalam *form* ini juga terdapat nomor faktur barang yang didasarkan dari tanggal pembeliaan barang.



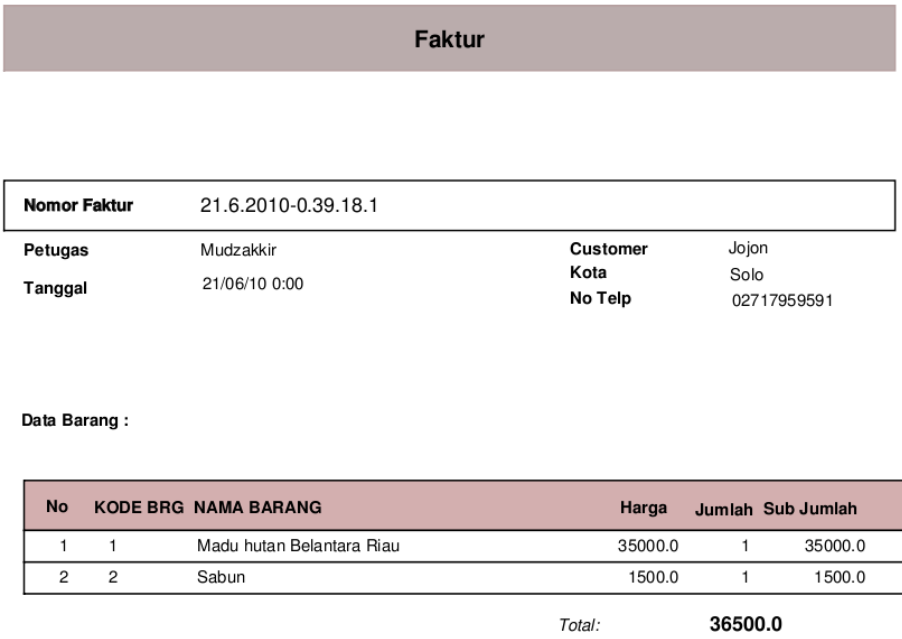
Gambar 4.6 *Form* Tambah Item Barang

Setelah data item barang suatu faktur disimpan ke dalam *database*, petugas dapat melihat daftar faktur di dalam daftar faktur.



Gambar 4.7 List Faktur

Petugas bisa melakukan proses edit dan hapus data faktur yang telah tersimpan di halaman daftar faktur. Selain itu petugas juga dapat mencetak data faktur yang diinginkan sebagai bukti penjualan.



Gambar 4.8 *Report* Faktur

4.1.2 Deskripsi *Class Diagram*

Petugas terdiri atas dua aktor, yaitu kasir dan admin. Petugas merupakan kelas induk dari pada kelas Admin dan kelas Kasir.

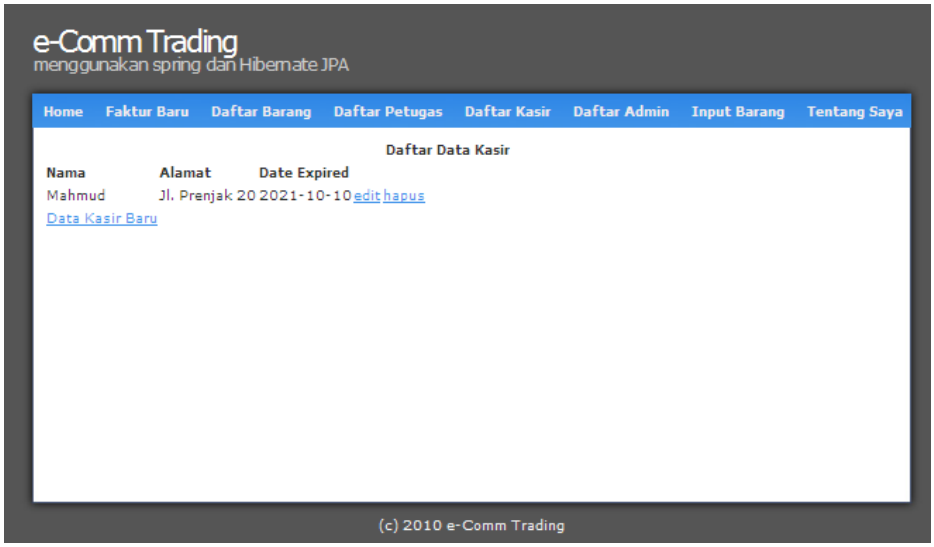
1. **Halaman Daftar Admin**

   Pada gambar daftar admin ini, petugas dapat melakukan manipulasi data dengan menambah, mengedit maupun menghapus daftar admin yang ada. Halaman admin hanya akan menampilkan daftar admin, sedangkan admin adalah petugas.

Gambar 4.9 Daftar Admin

2. Halaman Daftar Kasir

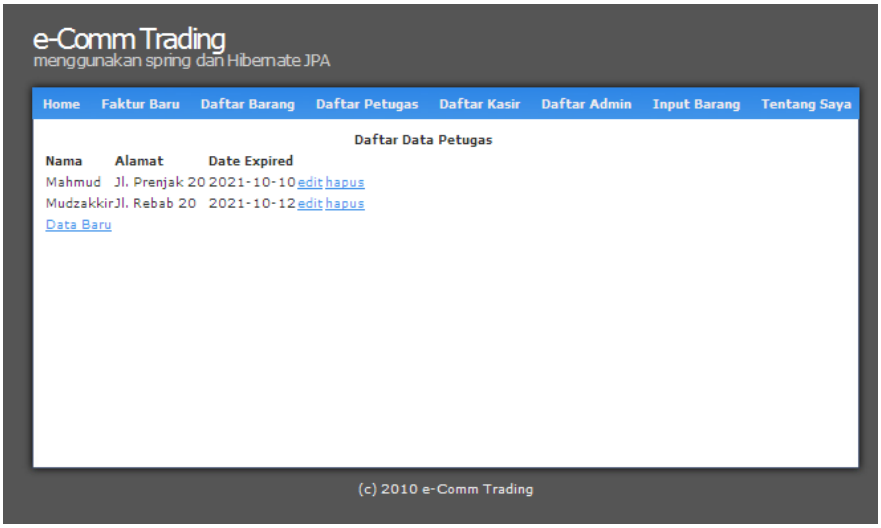
Petugas dapat melakukan manipulasi data dengan menambah, mengedit, maupun menghapus data kasir yang diinginkan pada halaman kasir. Halaman kasir hanya akan menampilkan daftar kasir, sedangkan kasir adalah petugas.



Gambar 4.10 Daftar Kasir

3. Halaman Daftar Petugas

Petugas dapat menambah, mengedit, maupun menghapus data yang ada di dalam data pada halaman petugas. Halaman petugas terdiri dari daftar kasir dan daftar admin, sehingga keseluruhan data kasir dan admin bisa dilihat pada daftar petugas.

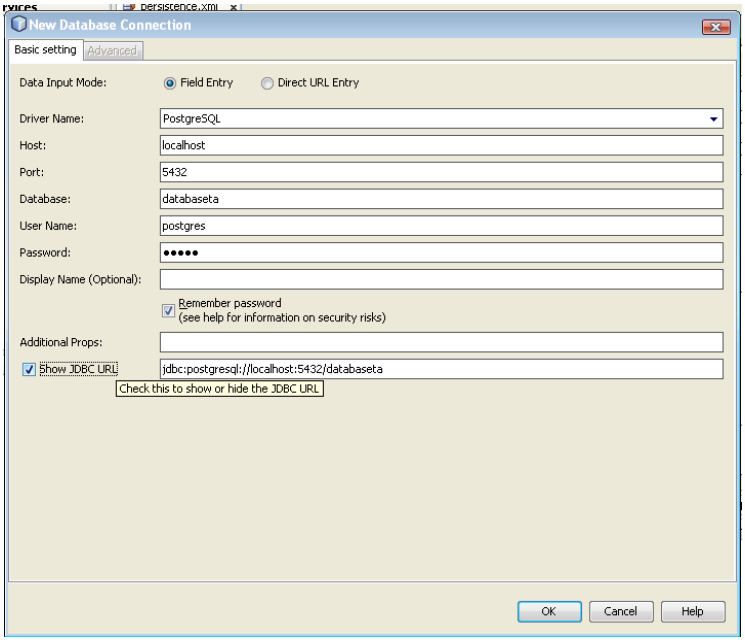


Gambar 4.11 Daftar Petugas

4.1.3 Implementasi Arsitektur *Three Tier* dan MVC

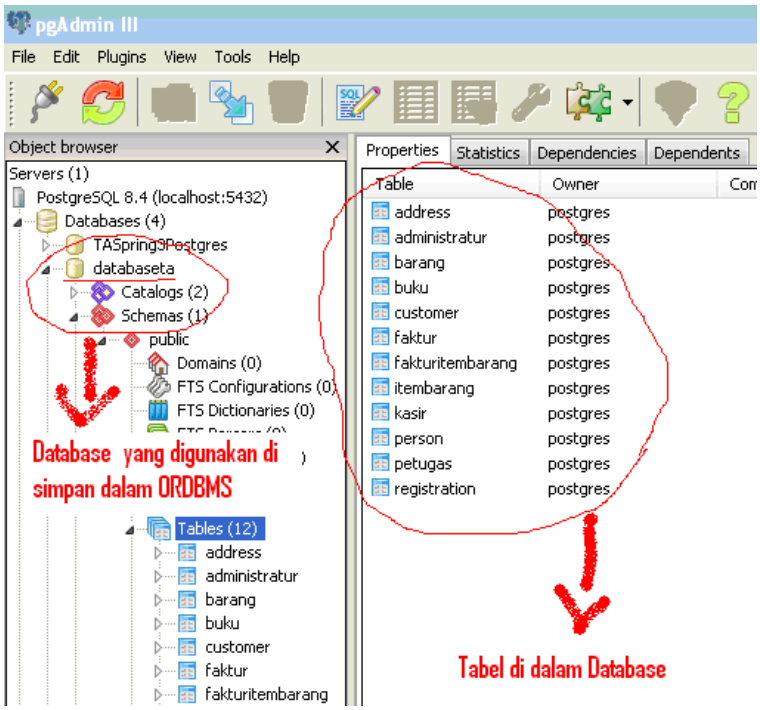
1. Implementasi Penggunaan ORDBMS PostgreSQL

Untuk menghubungkan antara aplikasi yang dibuat dengan ORDBMS PostgreSQL, diperlukan konfigurasi koneksi. Konfigurasi yang perlu diperhatikan adalah *database server* yang digunakan, *host*, *port*, nama *database*, *username database*, dan password *username*.



Gambar 4.12 Konfigurasi *Database*

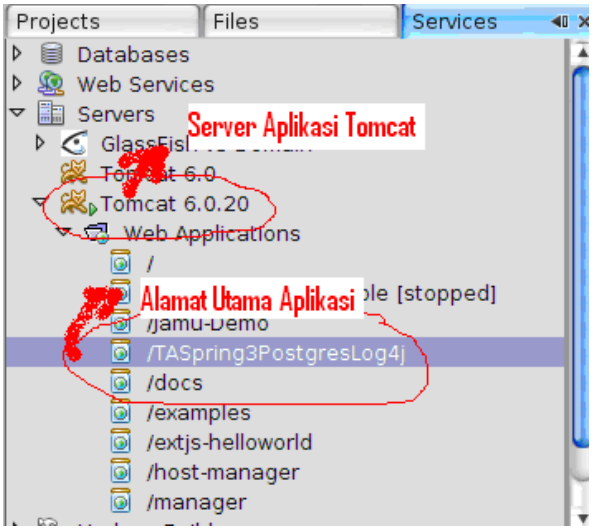
Hibernate JPA secara otomatis akan menciptakan tabel dalam *database* tanpa dibuat secara manual. *Database* perlu disiapkan, akan tetapi tabel-tabel tidak perlu dibuat secara manual.



Gambar 4.13 Implementasi Penggunaan PostgreSQL

2. Implementasi Tomcat sebagai *Server* Aplikasi

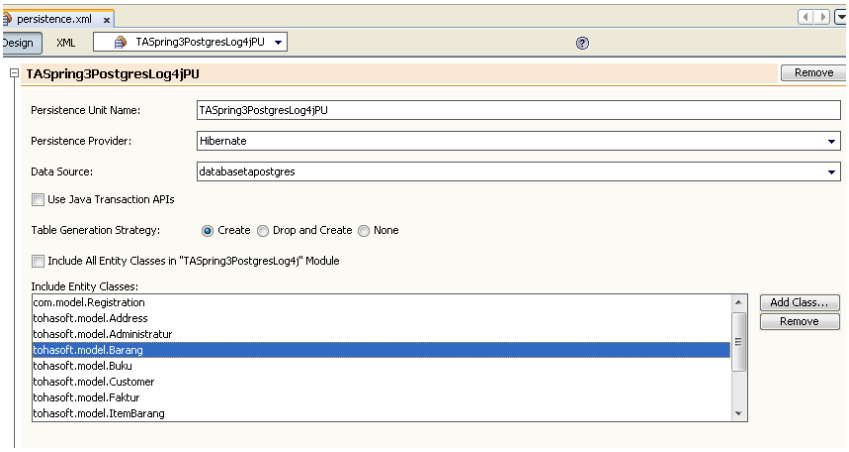
Aplikasi berjalan di atas aplikasi *server* tomcat. Alamat utama dari aplikasi berada di /TASpring3PostgresLog4j, sehingga diakses melalui alamat *host*:*port*/TASpring3PostgresLog4j.



Gambar 4.14 *Server* Aplikasi Tomcat

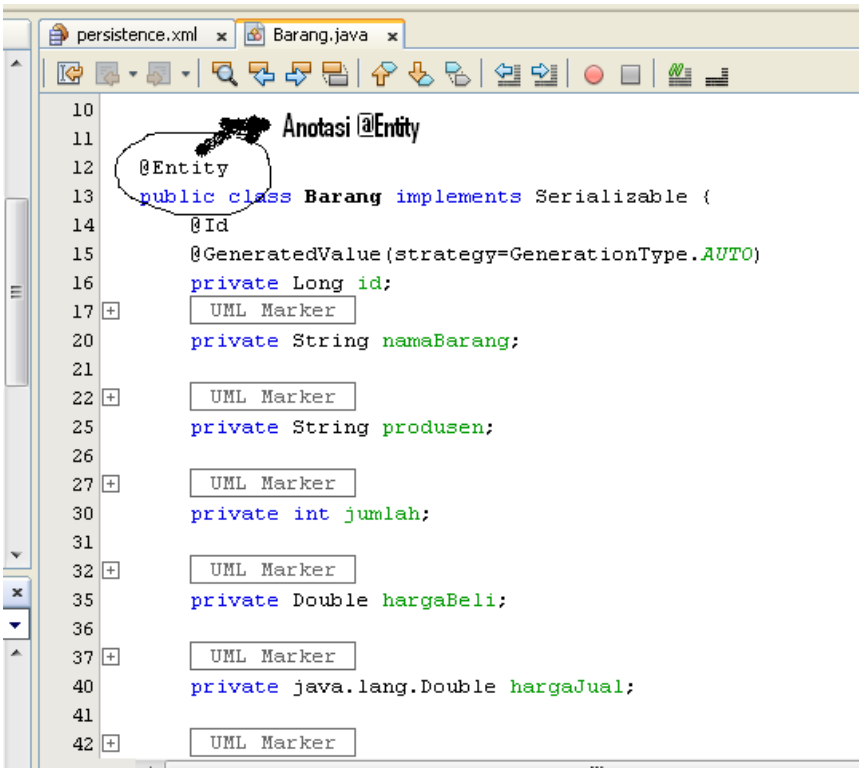
3. Implementasi Hibernate JPA sebagai Model

Hibernate JPA menggunakan konfigurasi persistece unit untuk menangani *beans* dari aplikasi.



Gambar 4.15 Persistence Unit

Entity model ditandai dengan annotasi @Entity untuk menandakan bahwa suatu kelas adalah sebuah *entity* model.



Gambar 4.16 Annotasi *Entity* Menandakan *Entity* Model

Entity model memerlukan id yang secara umum bertipe long, yang haruslah bersifat unik. Terdapat beberapa macam cara pemberian id, diantaranya adalah secara acak memberi nilai unik, dan auto increment.

Untuk me-*load* suatu *bean*, maka diperlukan DAO dari sisi model. DAO secara umum berbentuk sebuah *interface*, dan memiliki sebuah *class* yang akan mengimplementasikan DAO tersebut, akan tetapi ada juga yang memiliki konfigurasi yang agak berbeda.

```
5
6   package tohasoft.dao;
7
8   import java.util.List;
9   import tohasoft.model.Barang;
10
11  /**
12   *
13   * @author mudzakkir
14   */
15  public interface BarangDao {
16      public void create(Barang barang);
17      public void delete(Barang barang);
18      public Barang getById(Long id);
19      public List<Barang> getAllBarang();
20  }
```

Gambar 4.17 BarangDAO

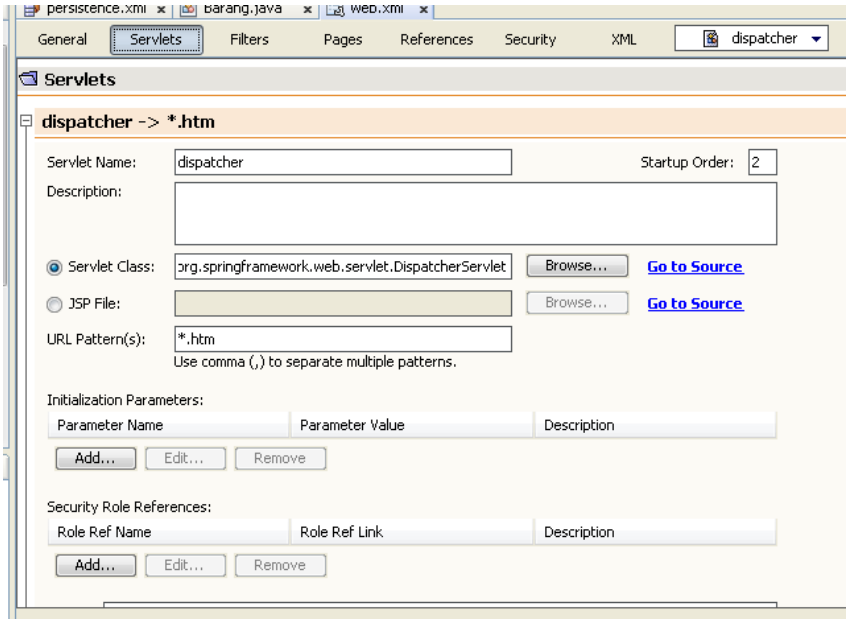
Spring MVC bisa tahu bahwa suatu kelas adalah suatu implementasi dari sebuah DAO dengan annotasi @Repository. Spring 3 sudah bisa mendeteksi suatu kelas adalah *repository*, tanpa mendeklarasikan *entity bean* di dalam file applicationContext. Hal tersebut adalah salah satu bentuk dari perpindahan dari teknlogi xml kepada annotasi yang diimplementasikan JEE versi 5.

```
19  */
20  @Repository
21  @Transactional
22  public class BarangImpl implements BarangDao{
23      @PersistenceContext
24      private EntityManager em;
25
27      public void create(Barang barang) {
28          em.merge(barang);
29      }
31      public void delete(Barang barang) {
32          em.remove(em.merge(barang));
33      }
35      public List<Barang> getAllBarang() {
36          return em.createQuery("from Barang").getResultList();
37      }
39      public Barang getById(Long id) {
40          return em.find(Barang.class, id);
41      }
42  }
```

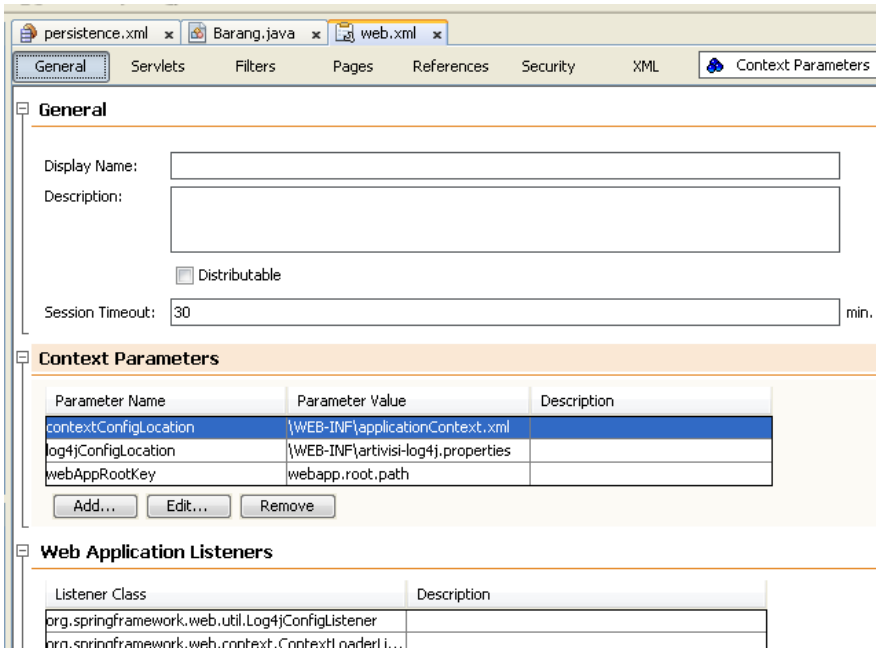
Gambar 4.18 Implementasi BarangDAO

4. Implementasi Spring MVC sebagai Controller

Spring MVC memerlukan 2 buah file konfigurasi utama, yaitu applicationContext dan dispatcherServlet. Tomcat akan mendeteksi keberadaan file tersebut dengan konfigurasi pada file web.xml.



Gambar 4.19 Konfigurasi Dispatcher Servlet web.xml

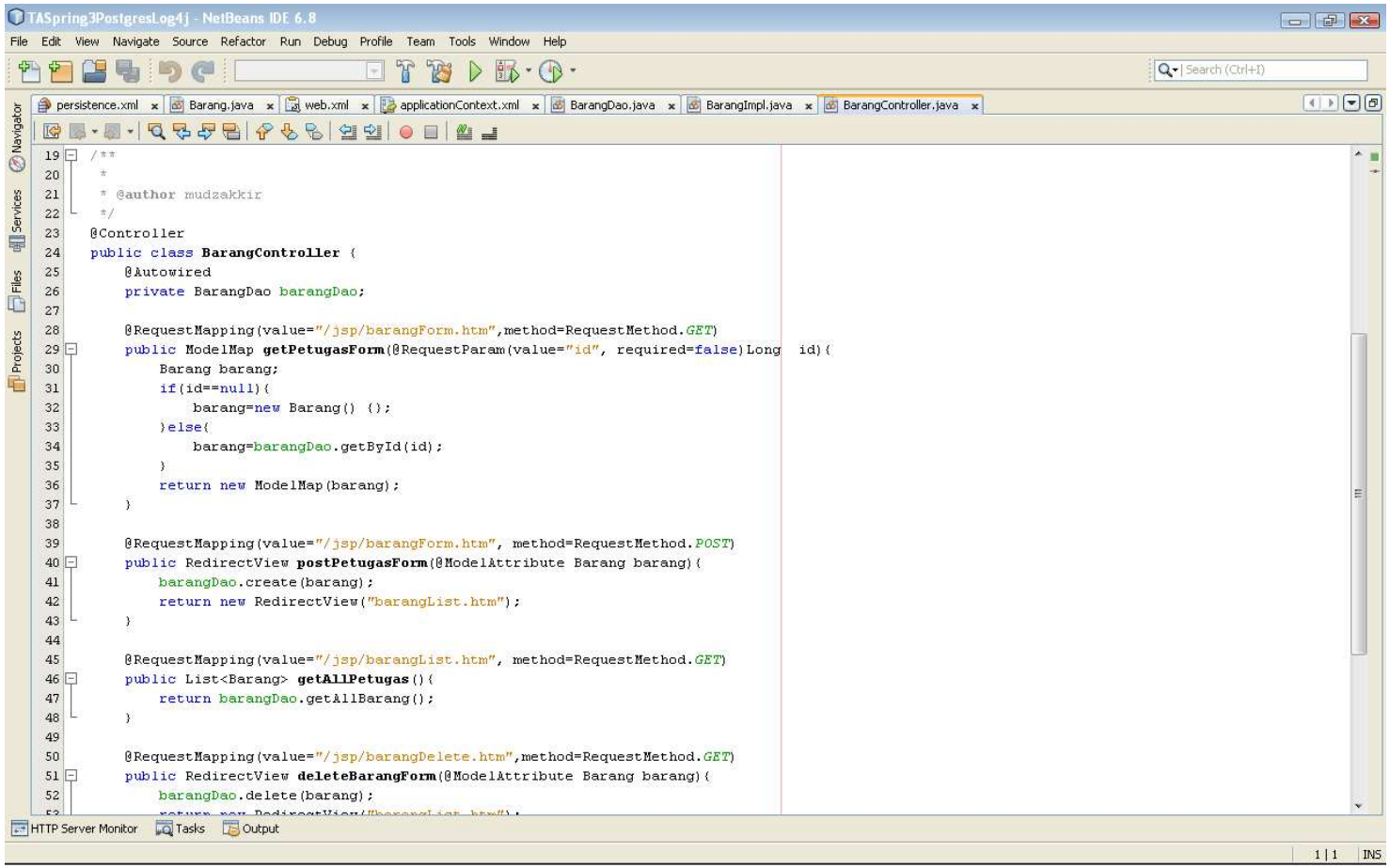


Gambar 4.20 Konfigurasi Application Context web.xml

Spring MVC memerlukan konfigurasi dalam application Context untuk memanajemen transaksi *entity bean*. Spring 3 sudah mampu mendeteksi suatu *repository bean*, sehingga tidak perlu mendeklarasikan semua *bean*.

Gambar 4.21 Konfigurasi *Entity* Manager Factory

Sebuah halaman memerlukan kontroller untuk dimanajemen. Ada beberapa macam jenis kontroller, diantaranya adalah MultiActionController, dan SimpleFormController.

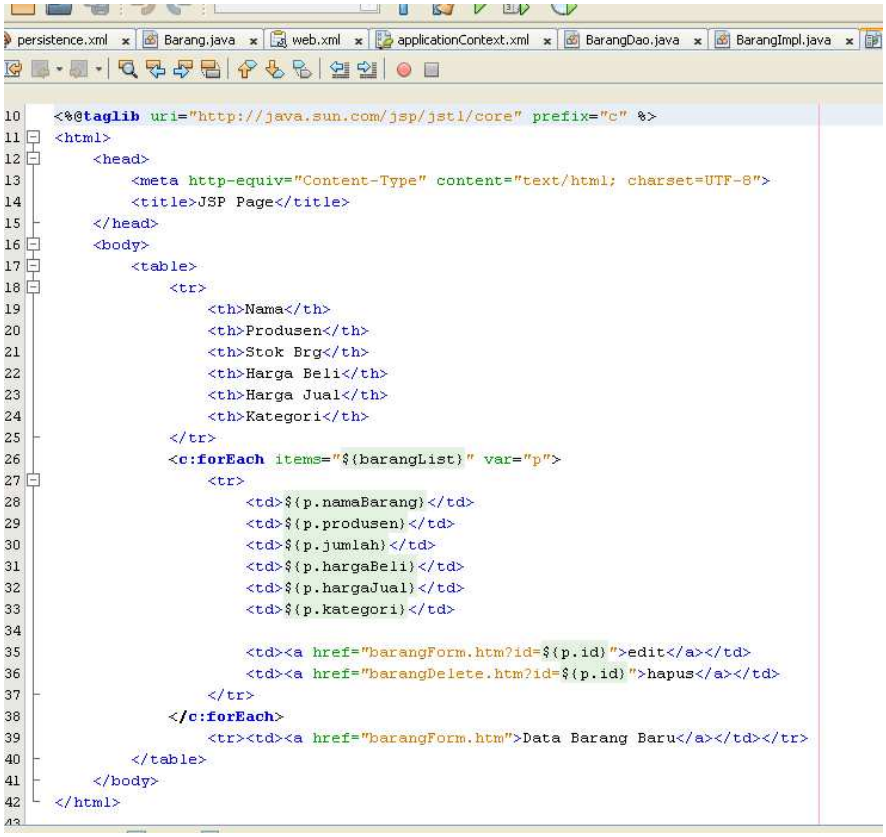


Gambar 4.22 *Controller* Halaman Manipulasi Model Barang

Controller halaman barang memerlukan DAO barang yang akan mengakses *bean* barang.

5. Implementasi JSP sebagai *View*

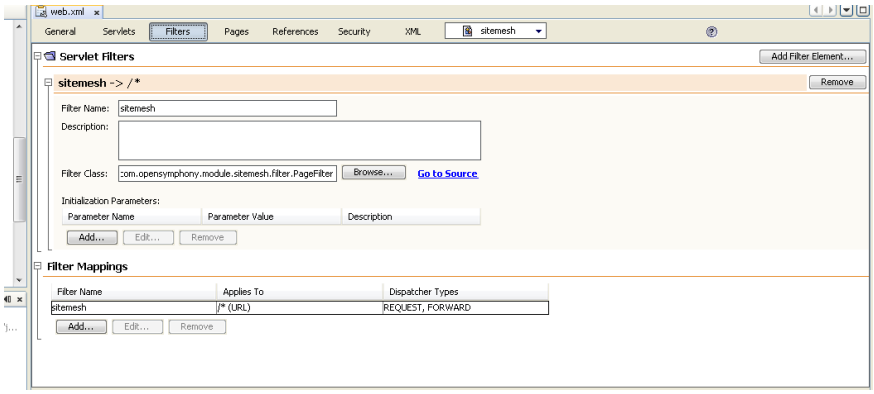
JSP menggunakan *taglib* dalam implementasinya. *Taglib* yang umumnya digunakan adalah jstl.



Gambar 4.23 Halaman Daftar Barang Contoh JSP

Sitemesh digunakan untuk *decorator templating*. Semua halaman nampak memiliki desain *view* yang serupa, akan tetapi di keseluruhan halaman tidak ada satupun kode yang me-*load* CSS, karena Sitemesh akan memberikan desain yang sama di semua halaman sebelum halaman ditampilkan di *view*.

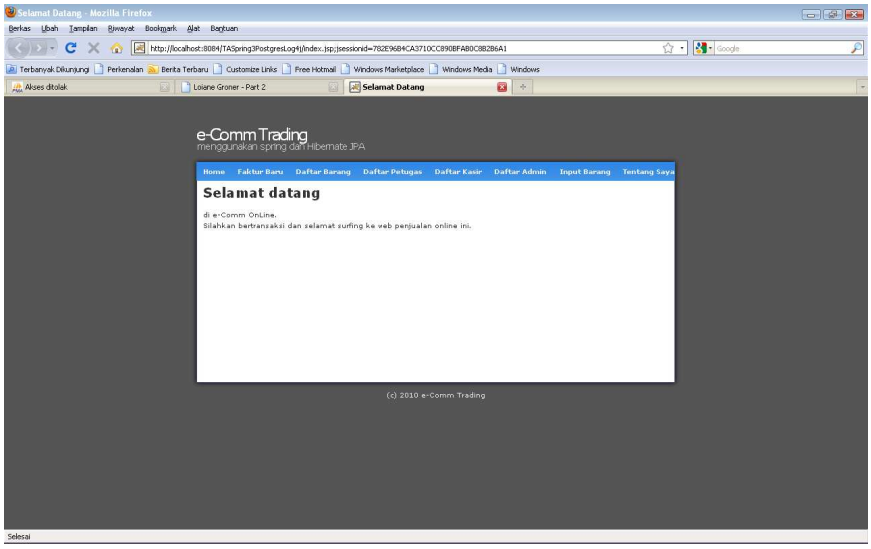
Konfigurasi sitemesh juga berada di web.xml.



Gambar 4.24 Konfigurasi Sitemesh

6. Implementasi *Thin Client* pada *Three Tier*

Client adalah thin *client* dalam arsitektur MVC, yaitu *client* yang tidak banyak memiliki business logic, dan kebanyakan hanya kode HTML biasa sebagai *view*. Aplikasi *web based* menggunakan *browser* sebagai *client*, sehingga aplikasi dibuka melalui *browser*.



Gambar 4.25 *Browser* sebagai *Client Interpreter*

4.2 Analisa

Dalam pembuatan program aplikasi dengan arsitektur MVC, selain kelebihan penulis menemukan beberapa kelemahan.

4.1.1 Kelebihan

a. Aplikasi ini lebih mudah untuk di manajemen dan di maintain.
b. Meminimalisir waktu dalam pembenahan kesalahan program.
c. Mudah dimengerti.

4.1.2 Kekurangan

a. Aplikasi ini sulit diimplementasikan oleh pemula.
b. Memerlukan tim yang ahli dalam pembuatan program MVC, sehingga aplikasi yang dihasilkan tidaklah aplikasi sembarangan.

BAB V
PENUTUP

5.1 Kesimpulan

Arsitektur MVC memisahkan antara *user interface*, model dalam *database*, dan *business logic*, sehingga mengurangi tingkat kerumitan suatu aplikasi. Hal ini akan sangat bermanfaat ketika *scope* aplikasi yang dibuat sangatlah luas, sehingga aplikasi yang dibuat dengan arsitektur MVC akan mudah untuk di-*maintain*.

5.2 Saran

Arsitektur MVC perlu dipertimbangkan untuk mengurangi biaya *maintenance* suatu aplikasi besar. Arsitektur MVC mampu mengurangi kerumitan dalam pembenahan aplikasi besar, seperti aplikasi bank, aplikasi peminjaman jangka panjang, dan aplikasi besar yang lain.

DAFTAR PUSTAKA

Ady, Martinus, 2007, *Membuat MasterDetail Report dengan iReport*, martinusadyh.web.id, 5 Mei 2010

Adams, Eriq, 2009, *Tutorial Membuat Aplikasi Katalog Buku menggunakan Spring dan Struts,* http://www.coderz.co.tv, 6 Februari 2010

Bima, Ifnu, 2008, *Arsitektur Three Tier dengan Swing, Spring, Hibernate, Jetty dan PostgreSQL*, ifnu.artivisi.com, 11 April 2010

Fleming, Matt, 2006, *Dynamic list binding in Spring MVC,* http://mattfleming.com, 1 April 2010

Jaglale, Jérôme, 2007, *Spring MVC Fast Tutorial*, maestric.com, 31 Maret 2010

Muhardin, Endy, 2009, *Konfigurasi lokasi logfile pada Spring MVC,* endy.artivisi.com, 11 April 2010

Prasetyo, Deny, 2010, *Spring By Example Contoh-contoh Spring Framework Komplit*, jasoet.wordpress.com, 27 Maret 2010

Sadewo, Kris, 2010, *Integrasi Spring MVC dan Hibernate JPA*, krissadewo.wordpress.com, 25 Maret 2010

Winterfeldt, David, 2008, *Spring by Example*, www.springbyexample.org, 21 Maret 2010